Creating elements

These two methods create new HTML elements which you can then insert into the document.

<table>
<tr><th>Selector</th><th>IE 5.5</th><th>IE 6</th><th>IE 7</th><th>IE8</th><th>IE9 pr3</th><th>FF 3.0</th><th>FF 3.5</th><th>FF 3.6</th><th>FF 4b1</th><th>Saf 4.0 Win</th><th>Saf 5.0 Win</th><th>Chrome 4</th><th>Chrome 5</th><th>Opera 10.10</th><th>Opera 10.53</th><th>Opera 10.60</th><th>Konqueror 4.x</th></tr>
<tr><td rowspan="2">createElement()
Create a new element</td><td colspan="5">yes</td><td colspan="4">yes</td><td colspan="2">yes</td><td colspan="2">yes</td><td colspan="3">yes</td><td>to be tested</td></tr>
<tr><td colspan="17">var x = document.createElement('P')
Create a new HTML element node <P> and temporarily place it in x, which is later inserted into the document.
• IE8 and below also support ('<P>').</td></tr>
<tr><td rowspan="2">createTextNode()
Create a new text node</td><td colspan="5">yes</td><td colspan="4">yes</td><td colspan="2">yes</td><td colspan="2">yes</td><td colspan="3">yes</td><td>to be tested</td></tr>
<tr><td colspan="17">var x = document.createTextNode('text')
Create a text node with content text and temporarily place it in x, which is later inserted into the document.</td></tr>
</table>

Getting elements

These methods are meant for getting the HTML elements you need from the document.

<table>
<tr><th>Selector</th><th>IE 5.5</th><th>IE 6</th><th>IE 7</th><th>IE8</th><th>IE9 pr3</th><th>FF 3.0</th><th>FF 3.5</th><th>FF 3.6</th><th>FF 4b1</th><th>Saf 4.0 Win</th><th>Saf 5.0 Win</th><th>Chrome 4</th><th>Chrome 5</th><th>Opera 10.10</th><th>Opera 10.53</th><th>Opera 10.60</th><th>Konqueror 4.x</th></tr>
<tr><td rowspan="2">getElementById()
Get the element with this ID</td><td colspan="3">almost</td><td colspan="2">yes</td><td colspan="4">yes</td><td colspan="2">yes</td><td colspan="2">yes</td><td colspan="3">yes</td><td>to be tested</td></tr>
<tr><td colspan="17">var x = document.getElementById('test')
Take the element with id="test" (wherever it is in the document) and put it in x.
If there is more than one element with id="test", the method selects the first in the document. All others are ignored.
• IE5-7 also return the element with name="test".</td></tr>
<tr><td rowspan="2">getElementsByClassName()
Get a nodeList of the elements with this class.</td><td colspan="4">no</td><td>yes</td><td colspan="4">yes</td><td colspan="2">yes</td><td colspan="2">yes</td><td colspan="3">yes</td><td>to be tested</td></tr>
<tr><td colspan="17">document.getElementsByClassName('test')
document.getElementsByClassName('test test2')
The first expression returns a nodeList with all elements that have a class value that contains "test". The</td></tr>
</table>

second one returns a nodeList will all elements that have a `class` value that contains both "test" and "test2" (in any order).

getElementsByTagName() Get all tags of this type	incomplete	yes				yes				yes		yes		yes			to be tested

```
var x = document.getElementsByTagName('P')
```

Make `x` into a nodeList of all P's in the document, so `x[1]` is the second P etc.

```
var x = y.getElementsByTagName('P')
```

Gets all paragraphs that are descendants of node `y`.

- The `*` argument, which ought to select all elements in the document, doesn't work in IE 5.5.
- Custom tags are not returned in Konqueror.

querySelectorAll() Get a nodeList of elements by CSS selector	no			yes		no	yes			yes		yes		yes			to be tested

```
document.querySelectorAll('.testClass')
document.querySelectorAll('.testClass + p')
```

Returns a nodeList with all elements that have a `class` value that contains "testClass"; or a nodeList with all paragraphs directly following such an element.
Essentially, this method allows you to use CSS syntax to retrieve elements.

Selector	IE 5.5	IE 6	IE 7	IE8	IE9 pr3	FF 3.0	FF 3.5	FF 3.6	FF 4b1	Saf 4.0 Win	Saf 5.0 Win	Chrome 4	Chrome 5	Opera 10.10	Opera 10.53	Opera 10.60	Konqueror 4.x

Node information

These four properties give basic information about all nodes. What they return depends on the node type. They are read-only, except fornodeValue.

There are three basic node types: element nodes (HTML tags), attribute nodes and text nodes. I test these properties for all these three types and added a fourth node type: the document node (the root of all other nodes).

Selector	IE 5.5	IE 6	IE 7	IE8	IE9 pr3	FF 3.0	FF 3.5	FF 3.6	FF 4b1	Saf 4.0 Win	Saf 5.0 Win	Chrome 4	Chrome 5	Opera 10.10	Opera 10.53	Opera 10.60	Konqueror 4.x
nodeName The name of the node in UPPER CASE	incomplete	yes				yes				yes		yes		yes			to be tested

```
x.nodeName
```

The name of node `x`. The correct names are:

Element	Attribute	Text	Comments	Document

the UPPER CASE tag name	the attribute name	`#text`	`#comment`	`#document`

- IE 5.5 doesn't support a `nodeName` for attributes and the document.
- IE 5.5 reports the `nodeName` of a comment as `!`.
- Konqueror doesn't see comment nodes.

nodeType The type of the node	incomplete	yes				yes				yes		yes		yes			to be tested

```
x.nodeType
```

The type of node `x`. The correct types are:

Element	Attribute	Text	Comments	Document
1	2	3	8	9

- IE 5.5: attributes and document not defined; comment has nodeType 1
- Konqueror doesn't see comment nodes.

Selector	IE 5.5	IE 6	IE 7	IE8	IE9 pr3	FF 3.0	FF 3.5	FF 3.6	FF 4b1	Saf 4.0 Win	Saf 5.0 Win	Chrome 4	Chrome 5	Opera 10.10	Opera 10.53	Opera 10.60	Konqueror 4.x
nodeValue The value of the node, if any. Read/write	incomplete	yes				yes				yes		yes		yes			to be tested

```
x.nodeValue
```

Get the value of node `x`

```
x.nodeValue = 'Test'
```

Set the value of node `x`

Element	Attribute	Text	Comments	Document
n/a	Value of attribute	Content of text node	Content of comment node	n/a

- IE 5.5 doesn't support `nodeValue` for attributes
- Konqueror doesn't see comment nodes.

tagName The tag name of an element node	almost					yes				yes		yes		yes			to be tested

```
x.tagName
```

Get the tag name of node `x`. Correct values are:

Element	Attribute	Text	Comments	Document
the UPPER CASE tag name	n/a	n/a	n/a	n/a

My advice is not to use `tagName` at all.

`nodeName` contains all functionalities of `tagName`, plus a few more. Therefore `nodeName` is always the better choice.

- In IE (all versions) the `tagName` of a comment node is `!`

Selector	IE 5.5	IE 6	IE 7	IE8	IE9 pr3	FF 3.0	FF 3.5	FF 3.6	FF 4b1	Saf 4.0 Win	Saf 5.0 Win	Chrome 4	Chrome 5	Opera 10.10	Opera 10.53	Opera 10.60	Konqueror 4.x

The DOM tree

Five properties and two arrays for walking through the DOM tree. Using these properties, you can reach nodes that are close to the current node in the document structure.

In general you shouldn't use too many of these properties. As soon as you're doing something like

```
x.parentNode.firstChild.nextSibling.children[2]
```

your code is too complicated. Complex relationships between nodes can suddenly and unexpectedly change when you alter the document structure, and altering the document structure is the point of the W3C DOM. In general you should use only one or two of these properties per action.

<table>
<tr><th>Selector</th><th>IE 5.5</th><th>IE 6</th><th>IE 7</th><th>IE8</th><th>IE9 pr3</th><th>FF 3.0</th><th>FF 3.5</th><th>FF 3.6</th><th>FF 4b1</th><th>Saf 4.0 Win</th><th>Saf 5.0 Win</th><th>Chrome 4</th><th>Chrome 5</th><th>Opera 10.10</th><th>Opera 10.53</th><th>Opera 10.60</th><th>Konqueror 4.x</th></tr>
<tr><td rowspan="2"><b>childNodes[]</b>
An array with all child nodes of the node</td><td colspan="5">yes</td><td colspan="4">yes</td><td colspan="2">yes</td><td colspan="2">yes</td><td colspan="3">yes</td><td>to be tested</td></tr>
<tr><td colspan="17"><code>x.childNodes[1]</code>
Get the second child node of node <code>x</code>.
The <code>childNodes</code> nodeList consists of all child nodes of the element, including (empty) text nodes and comment nodes.
• IE up to 8 does not count empty text nodes. Since this is the desired behaviour as far as I'm concerned, it still gets a Yes.
• Konqueror ignores comment nodes.</td></tr>
<tr><td rowspan="2"><b>children[]</b>
An array with all child element nodes of the node</td><td colspan="4">almost</td><td>yes</td><td>no</td><td colspan="3">yes</td><td colspan="2">yes</td><td colspan="2">yes</td><td colspan="3">yes</td><td>to be tested</td></tr>
<tr><td colspan="17"><code>x.children[1]</code>
Get the second element child node of node <code>x</code>.
Where <code>childNodes</code> holds all child nodes, <code>children</code> only holds those that are element nodes (HTML tags).</td></tr>
</table>

<table>
<tr><td></td><td colspan="17">• IE up to 8 incorrectly counts comment nodes, too.</td></tr>
<tr><td rowspan="2">firstChild
The first child node of the node</td><td colspan="5">yes</td><td colspan="4">yes</td><td colspan="2">yes</td><td colspan="2">yes</td><td colspan="3">yes</td><td>to be tested</td></tr>
<tr><td colspan="17">x.firstChild
Get the first child node of node x.
• Konqueror ignores comment nodes.</td></tr>
<tr><td rowspan="2">lastChild
The last child node of the node</td><td colspan="5">yes</td><td colspan="4">yes</td><td colspan="2">yes</td><td colspan="2">yes</td><td colspan="3">yes</td><td>to be tested</td></tr>
<tr><td colspan="17">x.lastChild
Get the last child of node x.
• Konqueror ignores comment nodes.</td></tr>
<tr><td>Selector</td><td>IE 5.5</td><td>IE 6</td><td>IE 7</td><td>IE8</td><td>IE9 pr3</td><td>FF 3.0</td><td>FF 3.5</td><td>FF 3.6</td><td>FF 4b1</td><td>Saf 4.0 Win</td><td>Saf 5.0 Win</td><td>Chrome 4</td><td>Chrome 5</td><td>Opera 10.10</td><td>Opera 10.53</td><td>Opera 10.60</td><td>Konqueror 4.x</td></tr>
<tr><td rowspan="2">nextSibling
The next sibling node of the node</td><td colspan="5">yes</td><td colspan="4">yes</td><td colspan="2">yes</td><td colspan="2">yes</td><td colspan="3">yes</td><td>to be tested</td></tr>
<tr><td colspan="17">x.nextSibling
Get the next child of the parent of x.
• Konqueror ignores comment nodes.</td></tr>
<tr><td rowspan="2">parentNode
The parent node of the node</td><td colspan="5">yes</td><td colspan="4">yes</td><td colspan="2">yes</td><td colspan="2">yes</td><td colspan="3">yes</td><td>to be tested</td></tr>
<tr><td colspan="17">x.parentNode
Get the parent node of x.</td></tr>
<tr><td rowspan="2">previousSibling
The previous sibling node of the node</td><td colspan="5">yes</td><td colspan="4">yes</td><td colspan="2">yes</td><td colspan="2">yes</td><td colspan="3">yes</td><td>to be tested</td></tr>
<tr><td colspan="17">x.previousSibling
Get the previous child of the parent of x.
• Konqueror ignores comment nodes.</td></tr>
<tr><td rowspan="2">sourceIndex
The index number of the node in the page source</td><td colspan="4">yes</td><td>incorrect</td><td colspan="4">no</td><td colspan="2">no</td><td colspan="2">no</td><td colspan="3">yes</td><td>to be tested</td></tr>
<tr><td colspan="17">x.sourceIndex
Get the sourceIndex of element x. This is also the index number for the element in the document.getElementsByTagName('*') array.</td></tr>
</table>

- IE returns a `sourceIndex` that’s one higher than in Opera. In IE8 and lower this refers to the correct element, but in IE9 it refers to the next element.

Selector	IE 5.5	IE 6	IE 7	IE8	IE9 pr3	FF 3.0	FF 3.5	FF 3.6	FF 4b1	Saf 4.0 Win	Saf 5.0 Win	Chrome 4	Chrome 5	Opera 10.10	Opera 10.53	Opera 10.60	Konqueror 4.x

DOM Traversal

A few useful properties that should have been in the DOM from the start but mysteriously weren’t.

Selector	IE 5.5	IE 6	IE 7	IE8	IE9 pr3	FF 3.0	FF 3.5	FF 3.6	FF 4b1	Saf 4.0 Win	Saf 5.0 Win	Chrome 4	Chrome 5	Opera 10.10	Opera 10.53	Opera 10.60	Konqueror 4.x
childElementCount The number of element children	no				yes	no	yes			yes		yes		yes			to be tested
	`x.childElementCount`																
firstElementChild The first child that is an element node	no				yes	no	yes			yes		yes		yes			to be tested
	`x.firstElementChild`																
lastElementChild The last child that is an element node	no				yes	no	yes			yes		yes		yes			to be tested
	`x.lastElementChild`																
nextElementSibling The next element node sibling	no				yes	no	yes			yes		yes		yes			to be tested
	`x.nextElementSibling`																
previousElementSibling The previous element node sibling	no				yes	no	yes			yes		yes		yes			to be tested
	`x.previousElementSibling`																
Selector	IE 5.5	IE 6	IE 7	IE8	IE9 pr3	FF 3.0	FF 3.5	FF 3.6	FF 4b1	Saf 4.0 Win	Saf 5.0 Win	Chrome 4	Chrome 5	Opera 10.10	Opera 10.53	Opera 10.60	Konqueror 4.x

Node manipulation

These five methods allow you to restructure the document. The average DOM script uses at least two of these methods.

The changes in the document structure are applied immediately, the whole DOM tree is altered. The browser, too, will immediately show the changes.

<table>
<tr><th>Selector</th><th>IE 5.5</th><th>IE 6</th><th>IE 7</th><th>IE8</th><th>IE9 pr3</th><th>FF 3.0</th><th>FF 3.5</th><th>FF 3.6</th><th>FF 4b1</th><th>Saf 4.0 Win</th><th>Saf 5.0 Win</th><th>Chrome 4</th><th>Chrome 5</th><th>Opera 10.10</th><th>Opera 10.53</th><th>Opera 10.60</th><th>Konqueror 4.x</th></tr>
<tr><td rowspan="2">appendChild()
Append a child node as the last node to an element</td><td colspan="5">yes</td><td colspan="4">yes</td><td colspan="2">yes</td><td colspan="2">yes</td><td colspan="3">yes</td><td>to be tested</td></tr>
<tr><td colspan="17">x.appendChild(y)
Make node y the last child of node x.
If you append a node that's somewhere else in the document, it moves to the new position.</td></tr>
<tr><td rowspan="2">cloneNode()
Clone a node</td><td colspan="5">yes</td><td colspan="4">yes</td><td colspan="2">yes</td><td colspan="2">yes</td><td colspan="3">yes</td><td>to be tested</td></tr>
<tr><td colspan="17">x = y.cloneNode(true | false)
Make node x a copy of node y. If the argument is true, the entire tree below y is copied, if it's false only the root node y is copied.
Later you insert the clone into the document.</td></tr>
<tr><td rowspan="2">insertBefore()
Insert a node into the child nodes of an element</td><td colspan="5">yes</td><td colspan="4">yes</td><td colspan="2">yes</td><td colspan="2">yes</td><td colspan="3">yes</td><td>to be tested</td></tr>
<tr><td colspan="17">x.insertBefore(y,z)
Insert node y as a child of node x just before node z.</td></tr>
<tr><td rowspan="2">removeChild()
Remove a child node from an element</td><td colspan="5">yes</td><td colspan="4">yes</td><td colspan="2">yes</td><td colspan="2">yes</td><td colspan="3">yes</td><td>to be tested</td></tr>
<tr><td colspan="17">x.removeChild(y)
Remove child y of node x.</td></tr>
<tr><td rowspan="2">replaceChild()
Replace a child node of an element by another child node</td><td colspan="5">yes</td><td colspan="4">yes</td><td colspan="2">yes</td><td colspan="2">yes</td><td colspan="3">yes</td><td>to be tested</td></tr>
<tr><td colspan="17">x.replaceChild(y,z)
Replace node z, a child of node x, by node y.</td></tr>
</table>

Text data

These methods are for manipulating text data, i.e. the contents of text nodes.

Selector	IE 5.5	IE 6	IE 7	IE8	IE9 pr3	FF 3.0	FF 3.5	FF 3.6	FF 4b1	Saf 4.0 Win	Saf 5.0 Win	Chrome 4	Chrome 5	Opera 10.10	Opera 10.53	Opera 10.60	Konqueror 4.x
appendData() Append data to a text node	no	yes		buggy		yes				yes		yes		yes			to be tested

```
x.appendData(' some extra text')
```

Appends the string `some extra text` to x, which must be a text node.

- IE8 and 9 appends the text, but doesn't show it until you click the element.

Selector	IE 5.5	IE 6	IE 7	IE8	IE9 pr3	FF 3.0	FF 3.5	FF 3.6	FF 4b1	Saf 4.0 Win	Saf 5.0 Win	Chrome 4	Chrome 5	Opera 10.10	Opera 10.53	Opera 10.60	Konqueror 4.x
data The content of a text node	yes					yes				yes		yes		yes			to be tested

```
x.data
```

The content of `x`, which must be a text node. The same as `x.nodeValue`.
Can also be set:

```
x.data = 'The new text'
```

Selector	IE 5.5	IE 6	IE 7	IE8	IE9 pr3	FF 3.0	FF 3.5	FF 3.6	FF 4b1	Saf 4.0 Win	Saf 5.0 Win	Chrome 4	Chrome 5	Opera 10.10	Opera 10.53	Opera 10.60	Konqueror 4.x
deleteData() Delete text from a text node	no	yes				yes				yes		yes		yes			to be tested

```
x.deleteData(4,3)
```

Delete some data from `x`, which must be a text node, starting at the fifth character and deleting three characters. Second argument is required.

Selector	IE 5.5	IE 6	IE 7	IE8	IE9 pr3	FF 3.0	FF 3.5	FF 3.6	FF 4b1	Saf 4.0 Win	Saf 5.0 Win	Chrome 4	Chrome 5	Opera 10.10	Opera 10.53	Opera 10.60	Konqueror 4.x
insertData() Insert text into a text node	no	yes				yes				yes		yes		yes			to be tested

```
x.insertData(4,' and now for some extra text ')
```

Insert the string `and now for some extra text` after the fourth character into `x`, which must be a text node.

Selector	IE 5.5	IE 6	IE 7	IE8	IE9 pr3	FF 3.0	FF 3.5	FF 3.6	FF 4b1	Saf 4.0 Win	Saf 5.0 Win	Chrome 4	Chrome 5	Opera 10.10	Opera 10.53	Opera 10.60	Konqueror 4.x
normalize() Merge adjacent text nodes into one node	no	yes				yes				yes		yes		yes			to be tested

```
x.normalize()
```

All child nodes of node `x` that are text nodes and have other text nodes as siblings, are merged. This is in fact the reverse of `splitText`: text nodes that were split, come together again.

Selector	IE 5.5	IE 6	IE 7	IE8	IE9 pr3	FF 3.0	FF 3.5	FF 3.6	FF 4b1	Saf 4.0 Win	Saf 5.0 Win	Chrome 4	Chrome 5	Opera 10.10	Opera 10.53	Opera 10.60	Konqueror 4.x
replaceData() Replace text in a text node `x.replaceData(4,3,' and for some new text ')` Replace three characters, beginning at the fifth one, of node `x`, which must be a text node, by the string `and for some new text`.	no	yes				yes				yes		yes		yes			to be tested
splitText() Split a text node into two text nodes `x.splitText(5)` Split the text node `x` at the 6th character. `x` now contains the first part (char. 0-5), while a new node is created (and becomes`x.nextSibling`) which contains the second part (char. 6-end) of the orginial text. • IE handles the first `splitText()` fine, but after you've normalized the text IE doesn't split it any more.	buggy					yes				yes		yes		yes			to be tested
substringData() Take a substring of the text in the text node `x.substringData(4,3)` Takes a substring of `x`, which must be a text node, starting at the fifth character and with a length of three characters. Thus it's the same as the old substr() method of strings.	no	yes				yes				yes		yes		yes			to be tested
wholeText The text of a text node *plus* the text in directly adjacent text nodes. Read only. This read-only property is useful if you want to get the entire text at a certain point and don't want to be bothered by borders between text nodes.	no				yes	no	yes			yes		yes		yes			to be tested
Selector	IE 5.5	IE 6	IE 7	IE8	IE9 pr3	FF 3.0	FF 3.5	FF 3.6	FF 4b1	Saf 4.0 Win	Saf 5.0 Win	Chrome 4	Chrome 5	Opera 10.10	Opera 10.53	Opera 10.60	Konqueror 4.x

Attributes

A bloody mess. Try influencing attributes in this order:

1. Try getting or setting a specific property, like `x.id` or `y.onclick`.
2. If there is no specific property, use getAttribute() orsetAttribute().
3. If even that doesn't work, try any other method or property in the table below. Most have horrible browser incompatibility patterns, though.
4. Avoid attributes[]. It's worse than anything else.

In my view any method or property concerning attribute nodes should also work on the `style` attribute, event handlers and custom attributes. If not I judge the method or property incomplete.

Selector	IE 5.5	IE 6	IE 7	IE8	IE9 pr3	FF 3.0	FF 3.5	FF 3.6	FF 4b1	Saf 4.0 Win	Saf 5.0 Win	Chrome 4	Chrome 5	Opera 10.10	Opera 10.53	Opera 10.60	Konqueror 4.x
attributes[index] An array with the attributes of a node, accessed by index number, in the order they're defined in the source code. Do not use Use `getAttribute()` instead	alternative			incorrect		incorrect				yes		yes		yes			to be tested

```
x.attributes[1]
```

This array consists of all defined attributes *in the source code order* .

- Firefox and IE8 try to create a list in source code order, but the order is off.
- IE5-7 takes the second *possible* attribute of node `x` (dataFld in the test), whether it's defined or not. Therefore it consists of all attributes that can *possibly* be defined on the node (84 all in all).
- IE 5.5 initially gives the value of the attribute; not the attribute object.

Do yourself a favour and don't use the indexed `attributes` array.

Selector	IE 5.5	IE 6	IE 7	IE8	IE9 pr3	FF 3.0	FF 3.5	FF 3.6	FF 4b1	Saf 4.0 Win	Saf 5.0 Win	Chrome 4	Chrome 5	Opera 10.10	Opera 10.53	Opera 10.60	Konqueror 4.x
attributes[key] An array with the attributes of a node, accessed by attribute name	incorrect	almost		yes		yes				yes		yes		yes			to be tested

```
x.attributes['align']
```

Get the align attribute object of node `x`. If the node has no align attribute, it returns `undefined` (except in IE, where it returns an attribute object that has no value.)

- IE5-7 doesn't return the value of a `style` attribute.
- IE 5.5 doesn't return custom attributes, and initially gives the attribute value instead of an attribute object.

Selector	IE 5.5	IE 6	IE 7	IE8	IE9 pr3	FF 3.0	FF 3.5	FF 3.6	FF 4b1	Saf 4.0 Win	Saf 5.0 Win	Chrome 4	Chrome 5	Opera 10.10	Opera 10.53	Opera 10.60	Konqueror 4.x
createAttribute() and setAttributeNode() Create a new attribute node and append it to an element node.	no	yes				yes				yes		yes		yes			to be tested

```
z = document.createAttribute('title');
z.value = 'Test title';
x.setAttributeNode(z)
```

This creates a title attribute with a value and sets it on node `x`.

Selector	IE 5.5	IE 6	IE 7	IE8	IE9 pr3	FF 3.0	FF 3.5	FF 3.6	FF 4b1	Saf 4.0 Win	Saf 5.0 Win	Chrome 4	Chrome 5	Opera 10.10	Opera 10.53	Opera 10.60	Konqueror 4.x
getAttribute() Get the value of an attribute	almost			yes		yes				yes		yes		yes			to be tested

```
x.getAttribute('align')
```

Gives the value of the align attribute of node `x`. Upper case attribute names are also allowed.

- In IE5-7, accessing the style attribute gives an object, and accessing the onclick attribute gives an anonymous function wrapped around the actual content.

getAttributeNode() Get an attribute node	no	almost		yes		yes				yes		yes		yes			to be tested

`x.getAttributeNode('align')`

Get the attribute object `align` of node `x`. This is an object, not a value.

- IE 6/7 don't allow you to access the value of `x.getAttributeNode('style')`.

hasAttribute() Check if a node has a certain attribute	no			yes		yes				yes		yes		yes			to be tested

`x.hasAttribute('align')`

Returns `true` when node `x` has an align attribute, `false` when it hasn't.

hasAttributes() Check if a node has attributes	no			yes		yes				yes		yes		yes			to be tested

`x.hasAttributes()`

Returns `true` when node `x` has attributes, `false` when it hasn't.

name The name of an attribute	no	yes				yes				yes		yes		yes			to be tested

`x.name`

The name of attribute node `x`.

Selector	IE 5.5	IE 6	IE 7	IE8	IE9 pr3	FF 3.0	FF 3.5	FF 3.6	FF 4b1	Saf 4.0 Win	Saf 5.0 Win	Chrome 4	Chrome 5	Opera 10.10	Opera 10.53	Opera 10.60	Konqueror 4.x
removeAttribute() Remove an attribute node	almost			yes		yes				yes		yes		almost			to be tested

`x.removeAttribute('align')`

Remove the `align` attribute from node `x`.

- IE5-7 and Opera don't remove event handlers.
- Konqueror doesn't remove the align attribute.

removeAttributeNode() Remove an attribute node	no	minimal	incomplete	yes		yes				yes		yes		almost			to be tested

```
x.removeAttributeNode(x.attributes['align'])
x.removeAttributeNode(x.attributes[1])
x.removeAttributeNode(x.getAttributeNode('align'))
```

Removes the attribute node. There is little difference with `removeAttribute()`, except in the argument.

- IE 6 does't remove anything, but doesn't give an error message either.
- IE 7 doesn't remove styles and event handlers.
- Opera doesn't remove event handlers.
- Konqueror doesn't remove the align attribute.

setAttribute() Set the value of an attribute	incomplete			yes		yes				yes		yes		yes			to be tested

`x.setAttribute('align','right')`

Set the align attribute of node `x` to `right`. The name and value are both strings.

- IE5-7 doesn't set styles and removes events when you try to set them.

setAttributeNode() See `createAttribute()`

value The value of an attribute	no	almost		yes		yes				yes		yes		yes			to be tested

`x.value`

The value of attribute `x`.

- IE6-7 gives `null` for style values.

Selector	IE 5.5	IE 6	IE 7	IE8	IE9 pr3	FF 3.0	FF 3.5	FF 3.6	FF 4b1	Saf 4.0 Win	Saf 5.0 Win	Chrome 4	Chrome 5	Opera 10.10	Opera 10.53	Opera 10.60	Konqueror 4.x

Miscellaneous

A lot of miscellaneous methods and properties that you'll rarely need. I use only two of them in an actual script.

Selector	IE 5.5	IE 6	IE 7	IE8	IE9 pr3	FF 3.0	FF 3.5	FF 3.6	FF 4b1	Saf 4.0 Win	Saf 5.0 Win	Chrome 4	Chrome 5	Opera 10.10	Opera 10.53	Opera 10.60	Konqueror 4.x
compareDocumentPosition() Gives the relative place of one element compared to another.	no				yes	yes				yes		yes		yes			to be tested

`x.compareDocumentPosition(y)`

Compares the document (DOM) position of element `y` to that of element `x` The method returns a bitmask:

- 1: Position disconnected
- 2: Precedes
- 4: Follows
- 8: Contains
- 16: Is contained by

All relevant numbers are added, and this sum is returned. So if `y` follows (4) and is contained by (16) `x`,

<table>
<tr><td></td><td colspan="17">the method returns 20.</td></tr>
<tr><td><b>contains()</b>
Check whether an element contains another element</td><td colspan="5">yes</td><td colspan="4">no</td><td colspan="2">yes</td><td colspan="2">yes</td><td colspan="3">yes</td><td>to be tested</td></tr>
<tr><td></td><td colspan="17">x.contains(y)
If node y is a descendant of node x, the method returns true, else false.
See this blog post for a Firefox workaround. (And yes, Firefox’s constant refusal to implement this very useful method is becoming extremely annoying.)</td></tr>
<tr><td><b>createDocument()</b>
Create a new document</td><td colspan="4">no</td><td>incorrect</td><td colspan="4">yes</td><td colspan="2">yes</td><td colspan="2">yes</td><td colspan="3">yes</td><td>to be tested</td></tr>
<tr><td></td><td colspan="17">x = document.implementation.createDocument('','',null)
Creates a new XML document.
• IE9 creates an HTML document.</td></tr>
<tr><td><b>createDocumentFragment()</b>
Create a document fragment</td><td>no</td><td colspan="4">yes</td><td colspan="4">yes</td><td colspan="2">yes</td><td colspan="2">yes</td><td colspan="3">yes</td><td>to be tested</td></tr>
<tr><td></td><td colspan="17">x = document.createDocumentFragment();
x.[fill with nodes];
document.[somewhere].appendChild(x);
Create a fragment, add a lot of nodes to it, and then insert it into the document. Note that the fragment itself is not inserted, only its child nodes.</td></tr>
<tr><td><b>documentElement</b>
The HTML tag</td><td colspan="5">yes</td><td colspan="4">yes</td><td colspan="2">yes</td><td colspan="2">yes</td><td colspan="3">yes</td><td>to be tested</td></tr>
<tr><td></td><td colspan="17">document.documentElement
Represents the root element of the XML document. In any HTML document, the <html> element is of course the root element.</td></tr>
<tr><td>Selector</td><td>IE 5.5</td><td>IE 6</td><td>IE 7</td><td>IE8</td><td>IE9 pr3</td><td>FF 3.0</td><td>FF 3.5</td><td>FF 3.6</td><td>FF 4b1</td><td>Saf 4.0 Win</td><td>Saf 5.0 Win</td><td>Chrome 4</td><td>Chrome 5</td><td>Opera 10.10</td><td>Opera 10.53</td><td>Opera 10.60</td><td>Konqueror 4.x</td></tr>
<tr><td><b>getElementsByName()</b>
Get elements by their name attribute</td><td colspan="5">incorrect and incomplete</td><td colspan="4">yes</td><td colspan="2">yes</td><td colspan="2">yes</td><td colspan="3">incorrect</td><td>to be tested</td></tr>
<tr><td></td><td colspan="17">var x = document.getElementsByName('test')
Create a nodeList with all elements that have name="test". It should ignore elements with id="test"
On my test page the <p>, <input>, <img> and <ppk> tags have this name, while there's also a paragraph</td></tr>
</table>

with `id="test"`. Ideally, all browsers should get the first four elements and ignore the fifth one.

- IE ignores the `<p>` and `<ppk>` tags with `name="test"`, but counts the `<div>` with `id="test"`
- Konqueror ignores the `<ppk>` tag.
- Opera counts the `<div>` with `id="test"`

Selector	IE 5.5	IE 6	IE 7	IE8	IE9 pr3	FF 3.0	FF 3.5	FF 3.6	FF 4b1	Saf 4.0 Win	Saf 5.0 Win	Chrome 4	Chrome 5	Opera 10.10	Opera 10.53	Opera 10.60	Konqueror 4.x
hasChildNodes() Check if the node has child nodes	yes					yes				yes		yes		yes			to be tested
	`x.hasChildNodes()` Returns `true` when node `x` has child nodes; `false` when it hasn't.																
item() Access an item in an array **Not necessary in JavaScript**	yes					yes				yes		yes		yes			to be tested
	`document.getElementsByTagName('P').item(0)` The same as `document.getElementsByTagName('P')[0]`. The `item()` method is meant for other programming languages where nodeLists like those returned by getElementsByTagName are not conveniently accessible as if they were arrays. You don't need `item()` at all in JavaScript.																
ownerDocument The document that 'owns' the element	no	yes				yes				yes		yes		yes			to be tested
	`x.ownerDocument` Refers to the document object that 'owns' node `x`. This is the document node.																

Microsoft extensions

As usual Microsoft has extended the standard somewhat. Though sometimes its extensions are brilliant (`innerHTML` springs to mind), in the case of the DOM Core they aren't.

Note the difference between W3C and Microsoft methods. The W3C methods are owned by the parent element of the node you want to adjust, the Microsoft methods by the node itself.

Selector	IE 5.5	IE 6	IE 7	IE8	IE9 pr3	FF 3.0	FF 3.5	FF 3.6	FF 4b1	Saf 4.0 Win	Saf 5.0 Win	Chrome 4	Chrome 5	Opera 10.10	Opera 10.53	Opera 10.60	Konqueror 4.x

Selector	IE 5.5	IE 6	IE 7	IE8	IE9 pr3	FF 3.0	FF 3.5	FF 3.6	FF 4b1	Saf 4.0 Win	Saf 5.0 Win	Chrome 4	Chrome 5	Opera 10.10	Opera 10.53	Opera 10.60	Konqueror 4.x
applyElement() Something with nodes	yes					no				no		no		no			to be tested
clearAttributes() Remove all attributes from a node	incomplete					no				no		no		no			to be tested
mergeAttributes() Copy all attributes of one node to another node	yes					no				no		no		no			to be tested
removeNode() Remove a node	yes					no				no		no		yes			to be tested
replaceNode() Replace a node by another node	yes					no				no		no		no			to be tested
swapNode() Swap two nodes	yes					no				no		no		no			to be tested

applyElement()

```
var y = document.createElement('i');
x.applyElement(y)
```

The `<i>` element is inserted into element `x`, around the text.

clearAttributes()

```
x.clearAttributes()
```

Remove all attributes from node `x`.

- IE doesn't clear event handlers and inline styles.

mergeAttributes()

```
x.mergeAttributes(y)
```

Copy all of node `y`'s attributes to node `x`.

removeNode()

```
x.removeNode(true | false)
```

Remove node `x` from the document. If you use the argument `true` its children are also removed; if you use `false` they aren't. Note that all text nodes count as children, too.

replaceNode()

```
x.replaceNode(y)
```

Replace node `x` by node `y`.

swapNode()

```
x.swapNode(y)
```

Test

Put node `x` in node `y`'s place and vice versa.